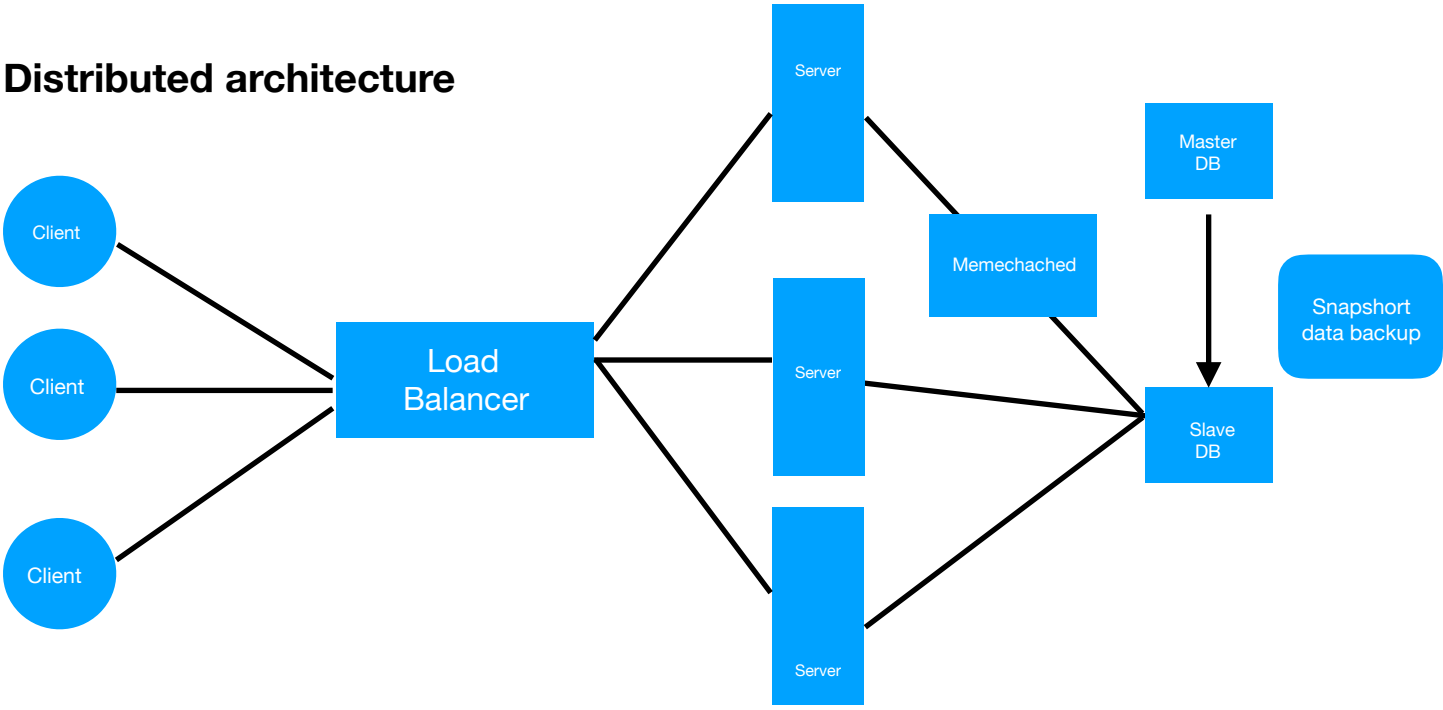
Distributed architecture



In the diagram above we have a load balancer that distributes the traffic across nodes. This architecture is a sticky session therefore once a socket connection is established to a server all subsequent messages are served from the same server. This is a cluster with 3 nodes and data is replicated in a slave database.

In this architecture, there are two database master and slave. The slave is a replica that is periodically updated from maser. We also have a backup snapshot from both the servers.

Sticky session

Sticky session can be achieved by two ways
1. Routing client based on their originated address.
2. Routing client based on cookie.

Data Replication and redundancy

The database has connection details, metadata and other details.
Assuming this is simple read and write, we consider one database as a master that receives all read and write request. The other DB is a replica. We achieved data redundancy and any loss of data.

Memcache

If there is a lot of read it is recommended to add a Memcache layer to offload heavy read database. In the diagram above the server can still write to the database but most commonly used data is retrieved from Memcache instead of connecting to the master database.

Fail-over

Load balancer makes the balancing decision over the first connection to select a server. We provide a mechanism for treating the subsequent connections as part of the same session. If the server is down, load balancer selects a new server.

Disconnection Detection:

TCP connection has keepalives check; however this not adequate and often does not work. It takes a very long time to detect the dead peer.
The recommended approach is a heartbeat mechanism check for every 10sec. If more than 3 attempt connection fails to mark the node is unreachable. There is a timeout for the session and update connection details accordingly.

Network failure is mostly a common problem for most failures. In case of connection, failure client has to establish a new connection. Any message in transit can be lost we can have an acknowledgment mechanism between client and server. It can be bidirectional if required.

Server response to reflect the distributed architecture:

Who: Number of connections

In this distributed architecture we store the connection details in the database. We are using a sticky connection therefore whenever a connection is established between a client and any node we update the details. Our heartbeat mechanism and timeout session keep connection information up to date.

Where: UUI

I have created IdGenerator.java file this is a distributed 64-bit unique id generator.

Timestamp in milliseconds 42 bits
Machine Id 10 bits
Counter 12 bits.